The Performance Evaluation Research Center (PERC)

David H. Bailey, LBNL

Participating Institutions:

Argonne Natl. Lab.	Univ. of California, San Diego
Lawrence Berkeley Natl. Lab.	Univ. of Illinois
Lawrence Livermore Natl. Lab.	Univ. of Maryland
Oak Ridge Natl. Lab.	Univ. of Tennessee, Knoxville

Website: http://perc.nersc.gov

The Performance Evaluation Research Center (PERC)

- An “Integrated Software Infrastructure Center” (ISIC) sponsored under DoE’s SciDAC program.
- Funding: approx. $2.4 million per year.
- Mission:
  - Develop a science of performance.
  - Engineer tools for performance analysis and optimization.
- Focus:
  - Large, grand-challenge calculations, especially SciDAC application projects.

Benefits to Computing Centers

Consider the economic value of improving the performance of a single high-end scientific application code by 20%.

Assume:

- $10 million computer system lease cost per year.
- $10 million per year in site costs, support staff, etc.
- 10-year lifetime of code.
- Code uses 5% of system cycles each year.

Savings: $2,000,000.

Scientific benefit (additional computer runs and research) is probably much higher.

Benefits to Vendors

- Large labs (like NERSC) rely heavily on commercial vendors for high-performance computer systems.
- We are invited by vendors to provide guidance on the detailed design of future systems.

BUT

- At present we can provide only vague information – little if any quantitative data or rigorous analysis.

The performance monitoring and modeling capability being developed in PERC will significantly improve our ability to influence future computer systems.

Performance Monitoring

- Flexible instrumentation systems to capture:
  - Hardware phenomena
  - Instruction execution frequencies
  - Memory reference behavior
  - Execution overheads
- An advanced data management infrastructure to:
  - Track performance experiments.
  - Collect data across time and space.
- User-friendly tools to tie performance data to user's source code.

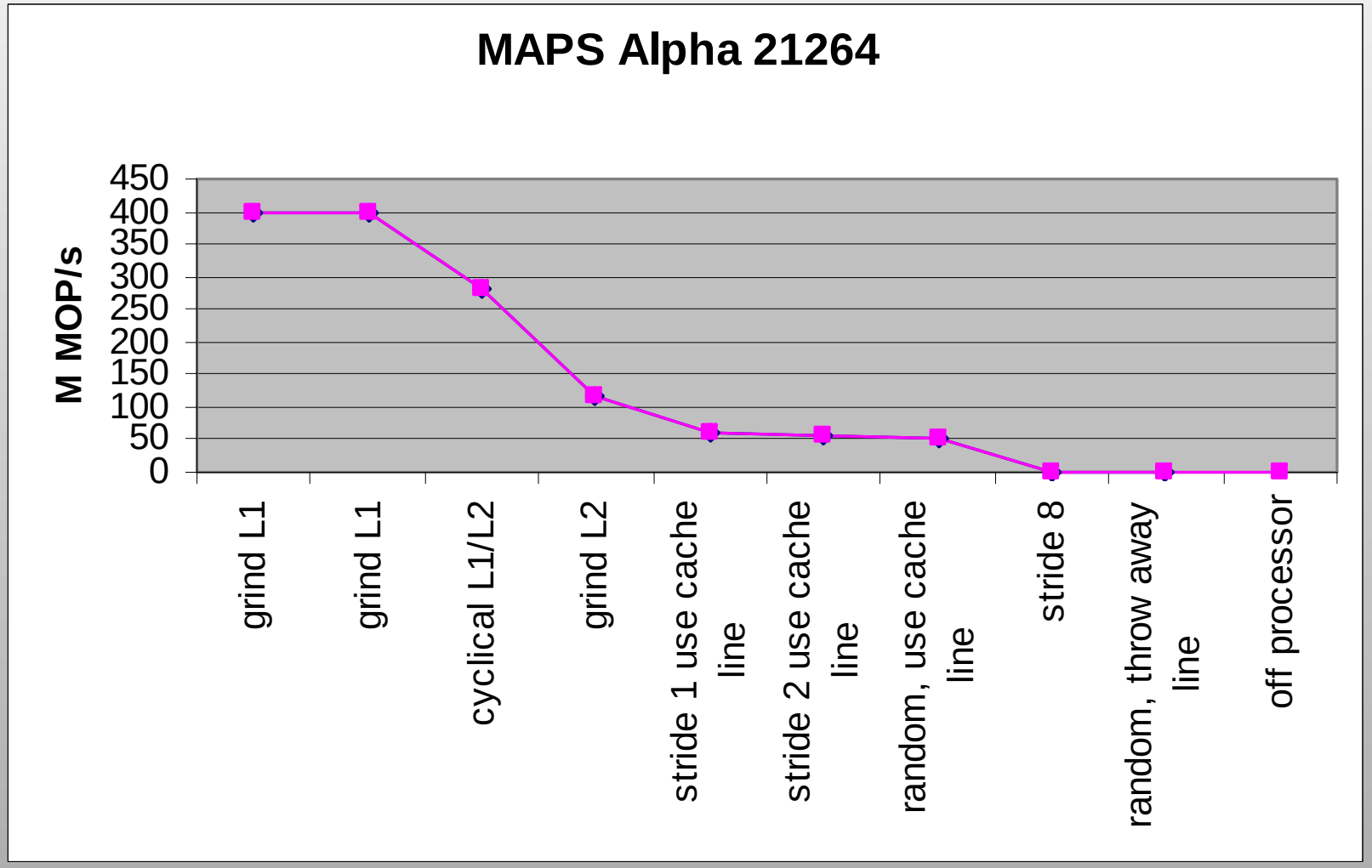
Performance Modeling

- *Application signature* tools characterize applications independent of the machine where they execute.
- *Machine signature* tools characterize computer systems, independent of the applications.
- *Convolution* schemes combine application and machine signatures to provide accurate performance models.
- Other performance modeling approaches:
  - Statistical models.
  - Phase models.
  - Performance bounds.

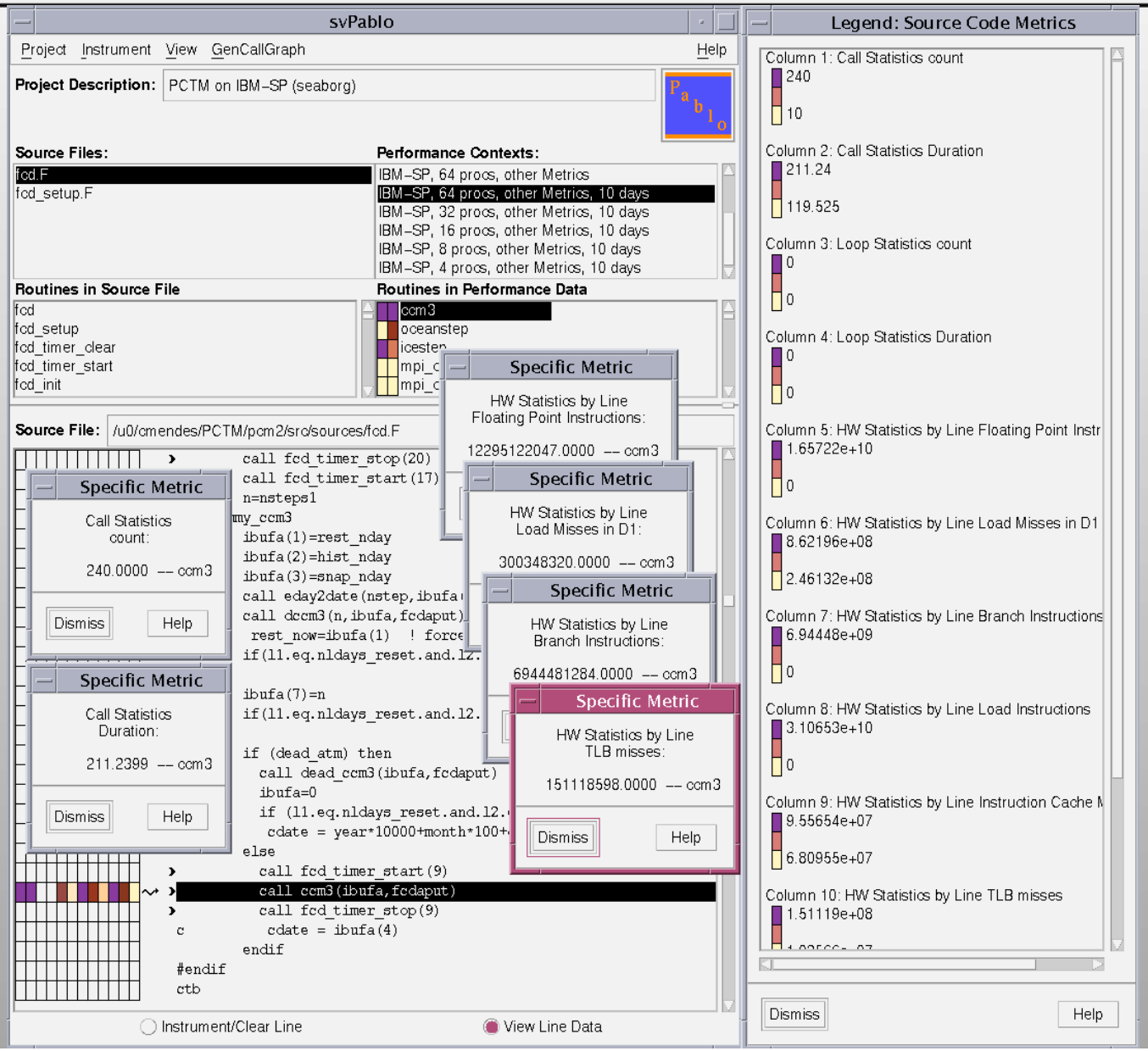
Performance Optimization

- *Compile-time optimization* mechanisms analyze source code to improve performance.
- *Self-tuning software* automatically tunes code based on real-time measurements of hardware environment.
- *Performance assertions* permit user-specified run-time tests to possibly change the course of the computation depending on results.
- *Performance portability* programming techniques to insure that code runs at near-optimal performance across a variety of modern systems.

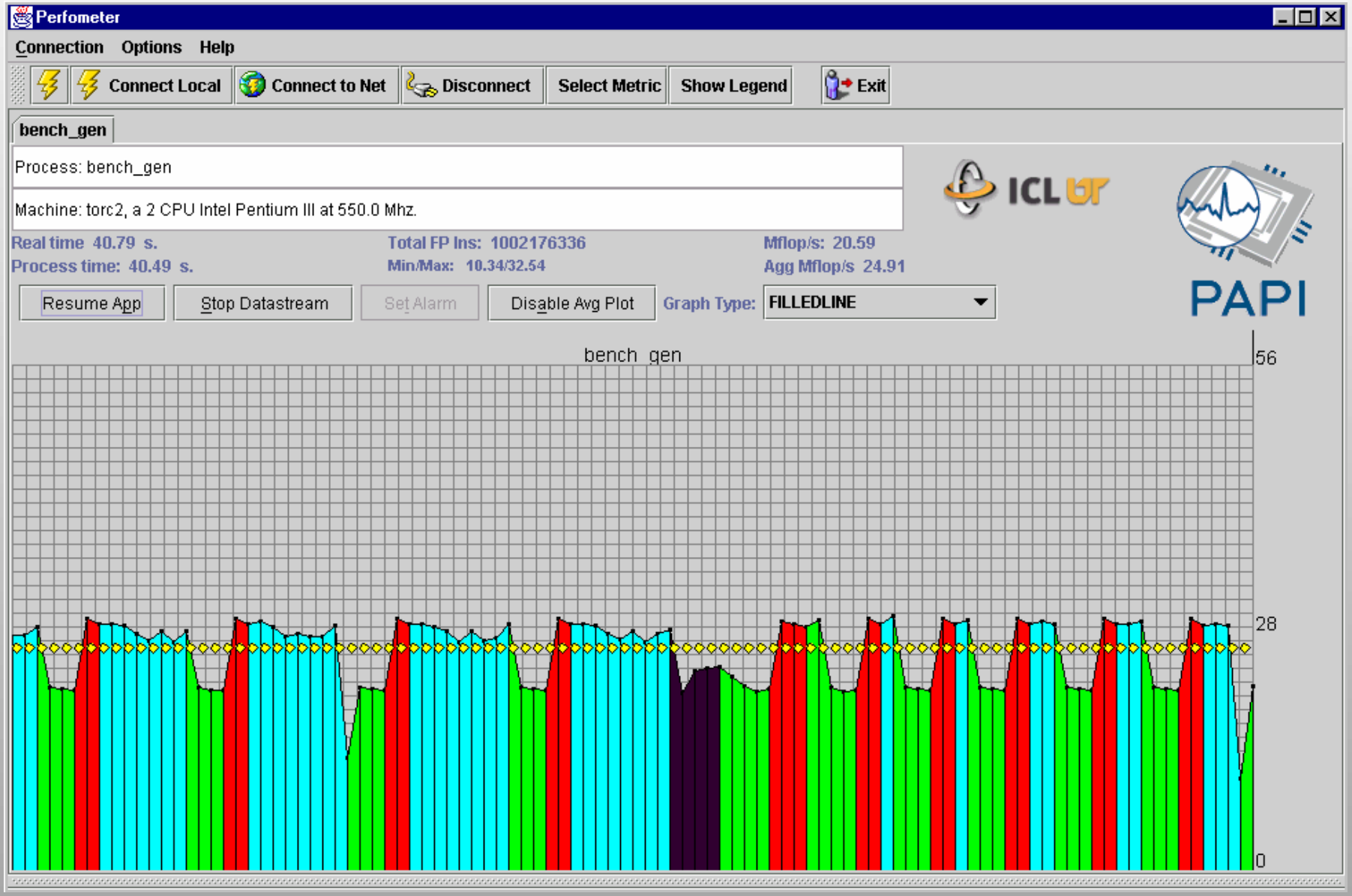
Measured Memory Access Patterns



SvPablo Graphical Interface



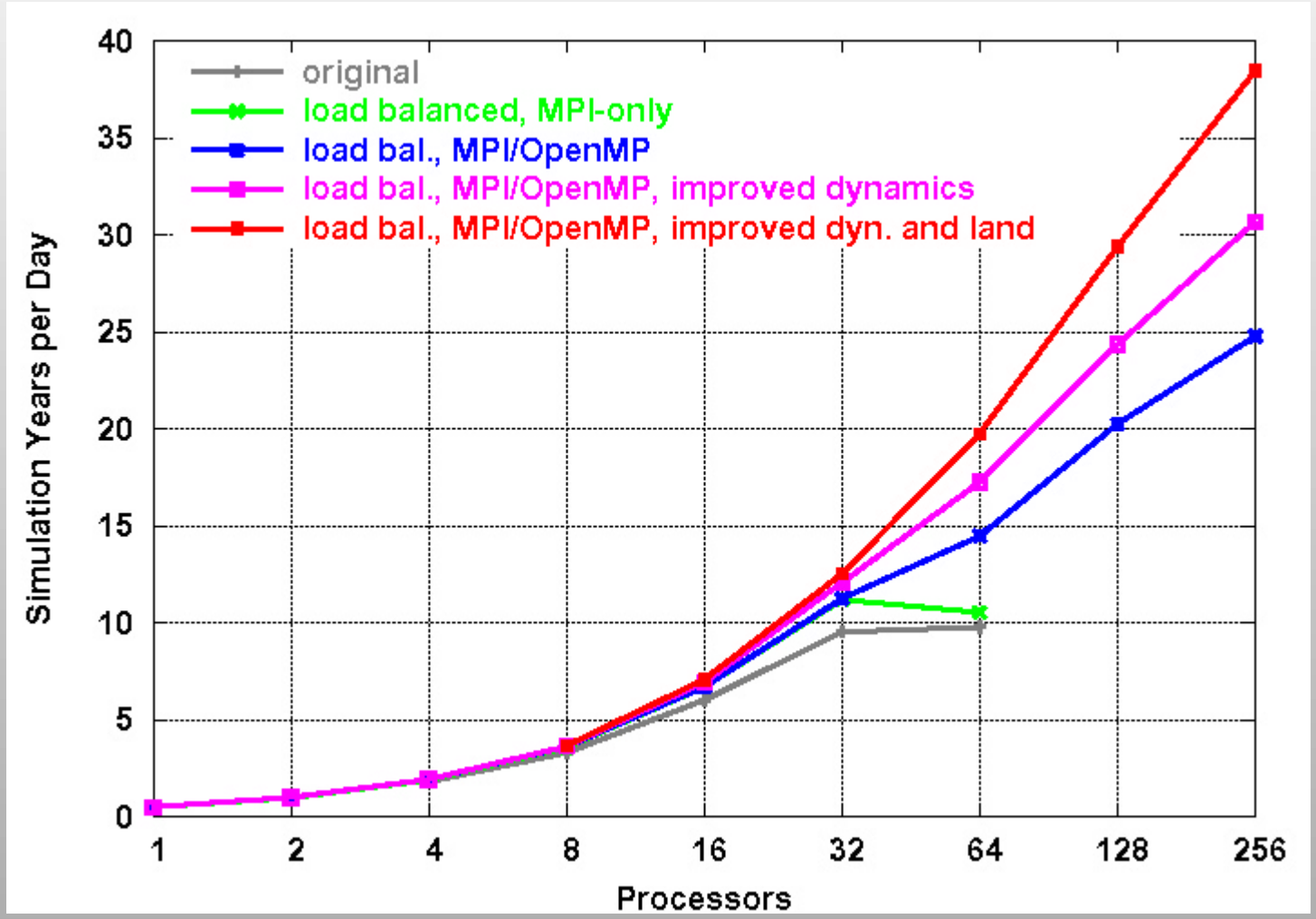
PAPI Perfometer Interface



Performance Signature Modeling
[A. Snavely, SDSC]

# CPUs	Real Time	Predicted Time	% Error
2	31.78	31.82	0.13
4	29.07	31.27	7.57
8	36.13	33.72	6.67
64	44.91	43.91	2.23
96	48.87	47.15	3.52
128	52.88	52.46	0.79

Climate Code Performance Improvement [P. Worley, ORNL]



Scalable Benchmarking
[LBNL/NERSC]

- Approach: Define a benchmark that automatically assesses performance over a broad range of relative distances between successive memory accesses.
- Feature: Independent of any particular cache structure or architecture.
- Result: A function graph of performance versus locality.

- Currently being developed by E. Strohmaier at LBNL.

Effective System Performance (ESP) Test [LBNL /NERSC]

- Analyzes system-level performance:
  - Efficiency of job scheduling system, applied to a mix of both small and large jobs.
  - Network contention in a fully-utilized system.
  - Job start-up times.
  - Time required for reboots (for example when restarting from regularly scheduled maintenance).
- Objective: Run a defined workload of varying system sizes in the best time possible, compared to the total capacity of the system.
- Measured scores range from 60% to 85%.

Sustained System Performance (SSP) Test [LBNL/NERSC]

- Developed at LBNL/NERSC for procurements.
- Defines sustained performance as the average performance of a suite of user application codes on say 256 CPUs.
- Defines sustained system performance by linear scaling to the total number of processors.
- Procurement SSP: Maximize the integral of SSP, measured at monthly intervals, over the lifetime of the contract.
- Provides strong value to the center.
- Provides flexibility to the vendor in meeting its contractual requirements.

Looking to the Future: The Massively Parallel Challenge

Systems featuring 10,000+ CPUs, present daunting challenges for performance analysis and tools:

- What performance phenomena should we measure?
- How can we collect and manage performance data spewed out by tens of thousands of CPUs?
- How can we visualize performance phenomena on 10,000+ CPUs?
- How can we identify bottlenecks in these systems?

Solution: Intelligent, highly automated performance tools, applicable over a wide range of system sizes and architectures, are needed.

Looking to the Future: Benchmarking and Modeling

- How can a center meaningfully procure a system with 10,000+ CPUs, a system 10 to 100 times more powerful than any system currently in existence?
- How can we define a benchmark that provides meaningful results for systems spanning four orders of magnitude in size?

- Reliable performance modeling techniques, usable with modest effort and expertise, offers the best hope for an solution here.

Looking to the Future: System Simulation

- "Computational scientists have become expert in simulating every phenomena except for the systems they run on." -- Speaker at Salishan 2003.
- System simulations heretofore have been used sparingly in system studies, because of the great cost and difficulty in parallelization of such simulations.

- Such simulations are now feasible, due to:
  - Availability of large-scale parallel systems.
  - Developments in the parallel discrete event simulation field.

Looking to the Future: User-Level Automatic Tuning

- Self-tuning software technology has already been demonstrated in a few large-scale libraries:
  - FFTW (MIT).
  - LAPACK-ATLAS (Univ. of Tennessee).

- Near term: Adapting these techniques to a wider group of widely used scientific libraries.
- Mid term: Automatically incorporate simple performance models into user application codes.
- Future goal: Automatically incorporate simple run-time tests, using compiler technology, into user application codes.

Working with PERC

For further information:

http://perc.nersc.gov